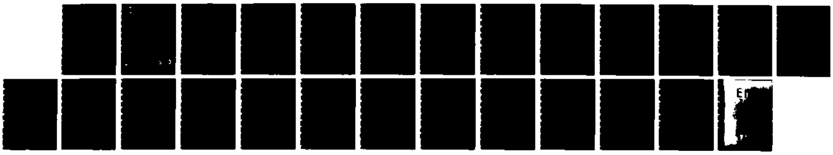

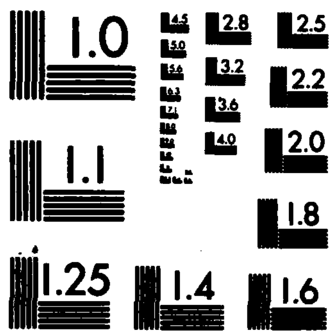

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

4

CAR-TR-36
CS-TR-1354

F49620-83-C-0082
December 1983

EMBEDDING OF NETWORKS OF PROCESSORS INTO HYPERCUBES

Angela Y. Wu

Center for Automation Research
University of Maryland
College Park, MD 20742

COMPUTER VISION LABORATORY

CENTER FOR AUTOMATION RESEARCH

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

(4)

CAR-TR-36 ✓ /
CS-TR-1354 ✓

F49620-83-C-0082
December 1983

EMBEDDING OF NETWORKS OF PROCESSORS INTO HYPERCUBES

Angela Y. Wu

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

The hypercube is a good host graph for the embedding of networks of processors because of its low degree and low diameter. Graphs such as trees and arrays can be embedded into a hypercube with small dilation and expansion costs, but there are classes of graphs which can be embedded into a hypercube only with large expansion cost or large dilation cost.

Research sponsored by the Air Force Office of Scientific Research (AFSC), under Contract F49620-83-C-0082. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The author also wishes to thank Azriel Rosenfeld for suggesting the problem, and Janet Salzman for preparing this paper.

1. Introduction

In the study of parallel computing, networks of processors are often organized into various configurations such as trees, pyramids, and mesh arrays [1-11]. These configurations can be represented as graphs. Using the properties and structures of the underlying graph effectively, the computation and communication speeds can often be improved.

A hypercube of degree d has 2^d nodes and each node has exactly d neighbors. The distance between any two nodes is less than or equal to d. Both the diameter of the hypercube and the degree of the nodes grow very slowly (logarithmically) with respect to the number of nodes in the hypercube. Therefore, the hypercube is a good configuration for networks of processors.

This paper studies the mapping of different network configurations into a hypercube network. In other words, it studies the one-to-one assocation of the processors of a network with the processors of a hypercube network and the costs of such mappings. The costs considered are the distances of images of adjacent processors and the size of the hypercube with respect to the number of processors in the given network. The process can be viewed as embedding of a graph (the underlying structure of the network) into a hypercube and the dilation and expansion costs of the embedding [12,13]. Graph embedding problems have applications in a wide variety of computational situations [13-18]. Embedding of graphs into trees and some issues of graph embedding have been studied by Rosenberg et al. in [13,19-21].

In this paper, the host graph in the embedding is always a hypercube. Section 2 defines and discusses some of the basic properties of hypercubes and graph embeddings. The rest of the paper discusses embedding of trees and graphs in hypercubes.

2. Hypercubes and graph embeddings

A hypercube of degree d has 2^d nodes and each node has d neighbors. The distance between any two nodes is less than or equal to d. The nodes in a hypercube may be labelled with binary numbers of length d. Two nodes $a_1a_2\ldots a_d$ and $b_1b_2\ldots b_d$ are adjacent to each other if and only if $\sum_{i=1}^{d} |a_i-b_i|=1$.

A degree d+1 hypercube C_{d+1} can be viewed as two degree d hypercubes $0C_d$ and $1C_d$ such that every node $0a_1a_2\ldots a_d$ in $0C_d$ is adjacent to one and only one node $1a_1a_2\ldots a_d$ in $1C_d$. Equivalently, given two degree d hypercubes, they can be combined into a degree d+1 hypercube by making the appropriate connections. Moreover, we can specify how the two hypercubes are to be combined as long as the conditions specified are consistent in the sense that it is possible to relabel one of the hypercubes so that all the neighbor relations are preserved. This is the same as applying a sequence of rigid transformations of rotations and reflections to a hypercube. For example, one can specify which node in $1C_d$ is to become the neighbor of the node A in $0C_d$ and also how the neighbors of A are to be mapped into neighbors of this node in $1C_d$.

Another property of the hypercube which is useful in later sections is that it is Hamiltonian. This can easily be proved by induction and using the fact that C_{d+1} is obtained from two C_d's with the appropriate connections.

An embedding f of a graph G=(V,E) in a graph G'=(V',E') is a one-to-one function f: $V \to V'$. The cost of an embedding f is $\max\{\text{distance}(f(A),f(B)) \mid (A,B)\in E\}$, i.e., the largest distance in

G' between the images of neighboring nodes of G. An embedding is said to preserve adjacency if its cost is 1. In the terminology of Rosenberg et al. [13] this is called the dilation cost. The expansion cost is the ratio of the number of nodes in V to the number of nodes in V'. Whenever no confusion can arise, we will refer to f as a function from G to G', and the image of a node A of G in G' will simply be identified as the node A in G'.

3. Embedding of binary trees in hypercubes

We first consider a complete binary tree where each non-leaf node has two children. A complete binary tree T_d of height d has 2^d-1 nodes. The smallest hypercube large enough to house T_d is of degree d. However, this embedding of T_d into C_d cannot preserve adjacency.

Proposition 1: A complete binary tree of height $d>2$ cannot be embedded in a hypercube of degree $\leq d$ such that adjacency is preserved. In other words, a complete binary tree cannot be embedded in a hypercube with a dilation cost of 1 and an expansion cost less than 2.

Proof: A hypercube of degree $<d$ has fewer nodes than T_d, hence T_d cannot be embedded. Consider a mapping f of nodes T_d into a hypercube C_d of degree d. Suppose the nodes of C_d are labeled with binary numbers between 0 and 2^d as in Section 2. Without loss of generality, we may assume that the root of T_d (at level 0) is mapped onto node 0 in C_d. Note that if the image of a level k node N of T_d has t 1's in its label, the images of node N's children at level k+1 have t+1 or t-1 1's in their labels. Since f(level 0 node) has no 1's in its label, the following are true:

(1) f(level i node) has no more than i 1's in its label.
(2) f(odd level node) has an odd number of 1's in its label.
(3) f(even level node) has an even number of 1's in its label.

Case 1: d is even.

The number of binary numbers of length d with an odd number of 1's is $\sum_{\text{odd } i} \binom{d}{i} = 2^{d-1}$.* The number of odd level nodes in T_d is $2+2^3+2^5+\ldots+2^{d-1} = \frac{2}{3}(2^d-1)$. But $2^{d-1} < \frac{2}{3}(2^d-1)$ if and only if $d>2$. This shows that if d is even and $d>2$, there are not enough nodes in C_d with an odd number of 1's in their labels to be the images of the odd level nodes of T_d.

Case 2: d is odd.

The number of binary numbers of length d with an even number of 1's is $\sum_{\text{even } i} \binom{d}{i} = 2^{d-1}$. The number of even level nodes in T_d is $2^0+2^2+2^4+\ldots+2^{d-1} = \frac{2^{d+1}-1}{3}$. But $2^{d-1} < \frac{2^{d+1}-1}{3}$ if and only if $d>1$. This shows that if d is odd and $d>1$, there are not enough nodes in C_d with an even number of 1's in their labels to be the images of the even label nodes of T_d. $\|$

For $d>2$, there is no adjacency-preserving embedding of T_d into C_d. However, Proposition 2 will show that if we allow a larger hypercube (one with twice as many nodes as necessary) as the target graph, then there is an adjacency preserving embedding of T_d into it. In other words, there is an embedding with a dilation cost of 1 and an expansion cost of approximately 2. On the other hand, Proposition 3 will show that there is an embedding of T_d into C_d with a dilation cost of 2 and an expansion cost of approximately 1.

* $(x+y)^r = \sum_{i=0}^{r} \binom{r}{i} x^i y^{r-i}$. Let $x=-1, y=1$; then we have $0=\sum_{i=0}^{r}\binom{r}{i}(-1)^i = \sum_{\text{even } i}\binom{r}{i} - \sum_{\text{odd } i}\binom{r}{i}$, and $\sum_{\text{even } i}\binom{r}{i} = \sum_{\text{odd } i}\binom{r}{i} = \frac{1}{2}\sum_{i=0}^{r}\binom{r}{i} = \frac{1}{2}\cdot 2^r = 2^{r-1}$.

Proposition 2: A complete binary tree T_d of height d (for d>0) can be embedded in a hypercube C_{d+1} of degree d+1, in such a way that the adjacencies of nodes T_d are preserved.

Proof: (By induction)

Figure 1 shows how T_1, T_2, and T_3 can be embedded in C_2, C_3, and C_4 with adjacency preserved (cost=1). Let f_i: $T_i \rightarrow C_{i+1}$ be an embedding of T_i into C_{i+1}. Suppose f_{d-1}: $T_{d-1} \rightarrow C_d$ preserves adjacency and (C_d, f_{d-1}) has the following property which we will refer to as the "free-free neighbor" property: $R=f_{d-1}$ (root of T_{d-1}) has a free neighbor A and A has a free neighbor $B \neq R$, i.e., $\{A,B\} \not\subseteq \{f_{d-1}(N) | N$ a node of $T_{d-1}\}$. Note that (C_4, f_3) has the "free-free neighbor" property.

Given a complete binary tree T_d of height d, its left subtree can be embedded by f_{d-1} in $0C_d$ of C_{d+1}, such that $0L=f_{d-1}$ (root) has a free neighbor 0A which has a free neighbor 0B. The right subtree can also be embedded in a degree d hypercube, call it C_d', using f'_{d-1} such that $R'=f'_{d-1}$ (root of right subtree) has a free neighbor A' which has a free neighbor B' in C'_d. We can now apply a rigid transformation T: $C'_d \rightarrow 1C_d$ such that T(R') is the neighbor of 0A in $1C_d$ and T(A') is the neighbor of 0B in $1C_d$. This is the same as forming a degree d+1 hypercube from two degree d hypercubes by making sure that they are oriented properly using rotation and reflection so that a node (R') and its neighbor (A') in one hypercube match two specific neighbor nodes (0A,0B) in the other hypercube. This can always be done since $d \geq 3$. Now the embedding f_d of T_d into C_{d+1} can be defined: $f_d(\text{root of } T_d) = 0A$, $f_d|\text{left subtree} = f_{d-1}$, $f_d|\text{right subtree} = T(f'_{d-1})$. Distances between 0A and the images of the left and right children of the root are both 1. Moreover, the free-free neighbor property is satisfied by (C_{d+1}, f_d) since 0A has a free neighbor 0B, and 0B has a free neighbor T(A'). See Figure 2. By induction, the above construction shows that a complete binary tree of height d ($d \geq 0$) can be embedded in a hypercube of degree d+1 with adjacency preserved. ||

Intuitively, the embedding is built up from the smaller subtrees. Each of the root's subtrees is embedded in one smaller hypercube, the root is mapped into a free neighbor of its left child's image, the right subtree's image hypercube is oriented in such a way that the image of the right subtree's root is a neighbor of the root's image. The free-free neighbor property ensures that a free neighbor is available in the embedding of a taller tree.

This embedding has the property that each subtree is embedded with adjacency preserved in a hypercube which has about twice the number of nodes of the subtree. The above construction also does not completely restrict how the other nodes of the two degree d hypercubes are to be matched.

<u>Proposition 3</u>: A complete binary tree T_d of height d ($d>0$) can be embedded in a hypercube C_d of degree d with cost = 2, i.e., neighbors in T_d are mapped into nodes of at most distance 2 away in C_d.

Proof: Figures 1a,b and Figure 3 show that T_1, T_2 and T_3 can be embedded in C_1, C_2, and C_3 in this way. Suppose T_{d-1} can be embedded in C_{d-1} by g_{d-1} and g_{d-1} (root of T_{d-1})=A and its left and right children are L,R; (C_{d-1}, g_{d-1}) has the cost 2 property, i.e., distance (A,L)=2 and distance (A,R)=1; and (g_{d-1}, C_{d-1}) also has the following property which we will refer to as the free neighbor property: the only free node in C_d is a neighbor of the root A. Figure 3 shows that (C_3, g_3) has cost 2 and the free neighbor property. We can embed T_d into C_d by first embedding the left subtree (height d-1) into the $0C_{d-1}$ hypercube with 0A being the root and 0B being the free neighbor of 0A, then embedding the right subtree (height d-1) into the $1C_{d-1}$ hypercube with 1A being the root, 1B being the free neighbor of 1A, and 0A,1A are neighbors, 0B,1B are neighbors; finally, the root is mapped to 1B. Clearly, distance (g_d(root),g_d(left child))=distance(1B,0A)=2, distance (g_d(root),g_d(right child))=1, the free node 0B is a neighbor of the root 1B. By induction, T_d can be embedded in C_d with cost=2. ||

The above construction shows that every node in T_d is mapped into a node distance 2 away from its left son's image and distance 1 away from its right son's image. We can also specify a symmetric embedding so that for d>3 the left and right subtrees of T_d are mapped into $0C_{d-1}$ and $1C_d$ in the same way so that the nodes in the same relative position in the left and right subtrees of any node are neighbors of each other. More specifically, we can use a string S of length i in $\{L,R\}^*$ to denote a label i node $(0<i<d)$ in T_d; this string S gives the path from the root to the node. For example, LR is a level 2 node which is the right child of the left child of the root. S_1LS_2 and S_1RS_2 $(S_1,S_2 \in \{L,R\}^*)$ are in the same relative positions in the left and right trees of the node denoted by S_1, and $g(S_1LS_2)$ and $g(S_1RS_2)$ are neighbors of each other. See Figure 4.

Let $A=a_1a_2\ldots a_i$, $B=b_1b_2\ldots b_i$ be the level i nodes in T_d, $a_j,b_j \in \{L,R\}, 1\le j\le i$. Suppose at positions $p_1\le p_2\le\ldots\le p_t$ $(t\le i)$ $a_{p_j}\ne b_{p_j}$; then $g(A)=g(a_1a_2\ldots a_{p_1-1}a_{p_1}a_{p_1+1}\ldots a_i)$, $g(a_1\ldots a_{p_1-1}b_{p_1}a_{p_1+1}\ldots a_{p_2-1}a_{p_2}a_{p_2+1}\ldots a_i)$, $g(a_1\ldots b_{p_1}\ldots a_{p_2-1}b_{p_2}a_{p_2+1}\ldots a_i)$, $\ldots$, $g(a_1\ldots b_{p_1}\ldots b_{p_2}\ldots b_{p_t}\ldots a_i)=g(B)$ is a path of length t in the hypercube C_d. Therefore, in this embedding the distance between two level i nodes is at most i.

The above propositions show how a complete binary tree can be embedded in a hypercube without greatly increasing the dilation cost or the expansion cost. If a binary tree is not complete, we can first complete the tree with imaginary nodes and then embed it in a hypercube. This in general is not economical because often a smaller hypercube can house the incomplete binary tree.

If we are interested in an adjacency preserving embedding of an incomplete binary tree, we can construct an embedding using the same principles as in the proofs of Propositions 2 and 3. We start with a leaf node at the highest level, map it into a hypercube of degree 1, and continue to put its ancestor nodes which has only one child in the same hypercube if possible. We go to a larger size hypercube if needed and always try to have the free neighbor property (as defined in the proof of Proposition 3) satisfied (the free-free neighbor property as defined in the proof of Proposition 2 if possible). When we reach a node A whose parent P has another child B, then we can try to put B's subtree in the same hypercube if it can be done easily, say, if it has only 3 or 4 nodes; otherwise, we embed B's tree in another hypercube, making sure that one of A or B's embeddings satisfies the free neigbor property (go to a higher dimensional hypercube if necessary). Then P can be mapped into the free neighbor in the hypercube with the free neighbor property. Of course, if one of the cubes has the free-free neighbor property, then P should be mapped into that cube. The free-free neighbor and the free neighbor properties ensure that there is a free node in the cube for the root of the taller subtree. In general, the hypercube (and embedding) that results from this method is smaller than the one obtained from completing the incomplete binary tree. For example, a subtree which is a single branch of length m can be embedded in a hypercube of degree $[\log_2 m]$ since hypercubes are Hamiltonian. The free-neighbor property is satisfied if the hypercube is of degree $\lceil \log_2 (m+1) \rceil$. The free-free neighbor property is satisfied if the cube is of degree $\lceil \log_2 (m+2) \rceil$.

4. k-ary trees

A k-ary tree can be turned into a binary tree by adding $\lceil \log_2 k \rceil - 1$ levels of nodes, i.e., at most $2^{\lceil \log_2 k \rceil} - 2$ nodes between a node and its k children. See Figure 5 for an example. A k-ary tree of height d becomes an equivalent binary tree of height $d + (d-1)(\lceil \log_2 k \rceil - 1) = \lceil \log_2 k \rceil d - \lceil \log_2 k \rceil + 1$. The number of new nodes needed is $2^{(\lceil \log_2 k \rceil d - \lceil \log_2 k \rceil + 1)} - 1$ - number of nodes in the k-ary tree. This distance between two adjacent nodes of the k-ary tree is $\lceil \log_2 k \rceil$ in the binary tree. This binary tree can be embedded in a hypercube as in Proposition 3 and we have proved the following:

Proposition 4: A k-ary tree K_d of height d can be embedded in a hypercube of degree $(d-1)\lceil \log_2 k \rceil + 1$ such that the distance between the images of adjacent nodes of K_d is at most $2 * \lceil \log_2 k \rceil$. ‖

If the k-ary tree K_d is complete and $k=2^m$ for some m then the equivalent binary tree (and thus the hypercube) has less than twice the number of nodes in K_d and the cost of the embedding is $2\lceil \log_2 k \rceil$ which is small in comparison with $\log_2$ (number of nodes in K_d). Again, if the k-ary tree is not complete, we can construct the embedding using the same principle as in the incomplete binary tree case and the images of the imaginary nodes added to form the equivalent binary tree can be used as images of real nodes in the k-ary tree.

5. Graphs

A graph G with n nodes can be embedded in a hypercube of degree $\lceil \log_2 n \rceil$ by simply associating nodes of the structures in any one-to-one fashion. This embedding uses the smallest size hypercube needed, but the dilation cost can be as high as $\lceil \log_2 n \rceil$. Another embedding method uses the spanning tree of the graph.

A graph $G=(V,E)$ of degree k and diameter d has a breadth-first spanning tree S which is k-ary with height d. An edge in E which is not an edge in S must connect only nodes in adjacent levels of S by the breadth first construction. Thus a graph can be considered as a k-ary tree with additional edges joining nodes whose levels differ by at most one.

For any graph G, we can construct an embedding by first constructing its breadth-first spanning tree S, extending this k-ary tree into a binary tree T, and then embedding the binary tree in a hypercube C using the method described in Section 3. This hypercube has degree $(d-1)\lceil \log_2 k \rceil+1$ and $2^{(d-1)\lceil \log_2 k \rceil+1}$ nodes. The cost of this embedding is $(d-1)\lceil \log_2 k \rceil+1$, since adjacent nodes in G can be mapped into two leaf nodes of T which are distance = (height of T-1) apart.

For a graph which is a k-ary tree with each leaf node connecting to k other leaf nodes, the costs of the embedding resulting from the above methods are almost identical and the cost is high. However, in both of these methods, the graph structure was not examined nor used. By examining the structure, one can usually get better embedding schemes with smaller costs.

For example, a $2^n \times 2^m$ array can be embedded in a hypercube of degree $n+m$ with adjacency preserved in a natural way: (a) First, embed each row i ($1 \le i \le 2^n$) in a hypercube $C_m^{(i)}$ of degree m; this can be done since hypercubes are Hamiltonian; (b) then combine $C_m^{(2j-1)}$ and $C_m^{(2j)}$ ($1 \le j \le 2^{n-1}$) into hypercube $C_{m+1}^{(j)}$ so that hypercube nodes representing array nodes in the same column are connected; then combine C_{m+1}^{2k-1} and C_{m+1}^{2k} ($1 \le k \le 2^{n-2}$) into hypercube C_{m+2}^{k}, again making sure that hypercube nodes representing array nodes at adjacent rows and same column are connected; continuing in this way, a hypercube C_{m+n} results and all adjacencies are preserved (cost=1). If the embedding methods for general graphs were used, the cost could be as high as $m+n$ because the spanning tree of the $2^n \times 2^m$ array is of height $n+m$ and quite skewed.

On the other hand, there are graphs that cannot be embedded into a hypercube with adjacency preserved no matter how large the hypercube is. For example, any graph which contains a cycle with length $2i+1$ for some $i>0$ cannot have an adjacency preserving (cost=1) embedding. A complete graph of n nodes is another example. However, we can have a cost 2 embedding if we use a hypercube of degree $n-1$. As a matter of fact, it is easy to see that any graph with n nodes can be embedded into a hypercube of degree $n-1$ so that adjacent nodes are mapped into nodes distance 2 apart.

6. Concluding remarks

The hypercube is a good host graph for the embedding of networks of processors because of its low degree and low diameter. Graphs such as these and arrays can be embedded into a hypercube with small costs. The design of the embedding mappings makes use of the structures of these graphs. In general, there is a trade-off between the dilation cost and the expansion cost (the size of the host hypercube). If this size is minimal, then the dilation cost of the embedding may be as large as log(number of nodes in the graph). On the other hand, there is always an embedding of an n node graph into a degree n-1 hypercube with a dilation cost of 2. We have also shown that there are classes of graphs which cannot have adjacency preserving (dilation cost=1) embeddings into hypercubes of any size.

References

1. M. J. B. Duff, "CLIP4: A large scale integrated circuit array parallel processor," Proc. Third IJCPR, 728-733.

2. K. E. Batcher, "Design of a massively parallel processor," IEEE Trans. Computers 29, 1980, 836-840.

3. W. Bouknight, et al., "The ILLIAC IV System," Proc. IEEE 60, 1972, 369-388.

4. A. P. Reeves, "A systematically designed binary array processor," IEEE Trans. Computers 29, 1980, 278-287.

5. D. Kuck, "A survey of parallel machine organization and programming," ACM Computing Surveys 9, 1977, 29-59.

6. L. Uhr, "Layered 'recognition cone' networks that preprocess, classify and describe," IEEE Trans. Computers 21, 1972, 758-768.

7. C. R. Dyer, "A quadtree machine for parallel image processing," Technical Report KSL 51, University of Illinois at Chicago Circle, 1981.

8. S. Tanimoto, "Towards hierarchical cellular logic: design considerations for pyramid machines," Department of Computer Science Technical Report 81-02-01, University of Washington, 1981.

9. A. R. Hanson and E. M. Riseman, "Processing Cones: A computational structure for image analysis," in S. Tanimoto and A. Klinger, eds., Structured Computer Vision, Academic Press, NY, 1980.

10. A. Wu and A. Rosenfeld, "Cellular graph automata (I and II)," Information and Control 42, 305-329, 330-353, 1979.

11. M. C. Pease III, "The indirect binary n-cube microprocessor array," IEEE Trans. Computers 26, 458-473, 1977.

12. A. L. Rosenberg, "Data encodings and their roots," Acta Informatica 9, 273-292, 1978.

13. J. W. Hong, K. Mehlhorn and A. L. Rosenberg, "Cost trade-offs in graph embeddings, with applications," JACM 30, 709-728, 1983.

14. R. A. DeMille, S. C. Eisenstat and R. J. Lipton, "On small universal data structures and related combinatorial problems," Proc. Johns Hopkins Conf. on Information Sciences and Systems, Baltimore, MD, 408-411, 1978.

15. R. J. Lipton, S. C. Eisenstat and R. A. DeMille, "Space and time hierarchies for collections of control structures and data structures," JACM 23, 720-732, 1976.

16. L. G. Valient, "Universality considerations in VLSI circuits," IEEE Trans. Computers 30, 135-140, 1981.

17. H. T. Kung and D. Stevenson, "A software technique for reducing the routing time on a parallel computer with a fixed interconnection network," High Speed Computer and Algorithm Optimization, Academic Press, NY, 423-433, 1977.

18. A. L. Rosenberg, "Three dimensional VLSI: A case study," JACM 30, 397-416, 1983.

19. A. L. Rosenberg, "Encoding data structures in trees," JACM 26, 668-689, 1979.

20. J. W. Hong and A. L. Rosenberg, "Graphs that are almost binary trees," SIAM J. Computing 11, 227-242, 1982.

21. A. L. Rosenberg, "Issues in the study of graph embeddings," Graph-theoretic Concepts in Computer Science, H. Noltemeier, ed., Springer-Verlag, NY, 150-176, 1981.

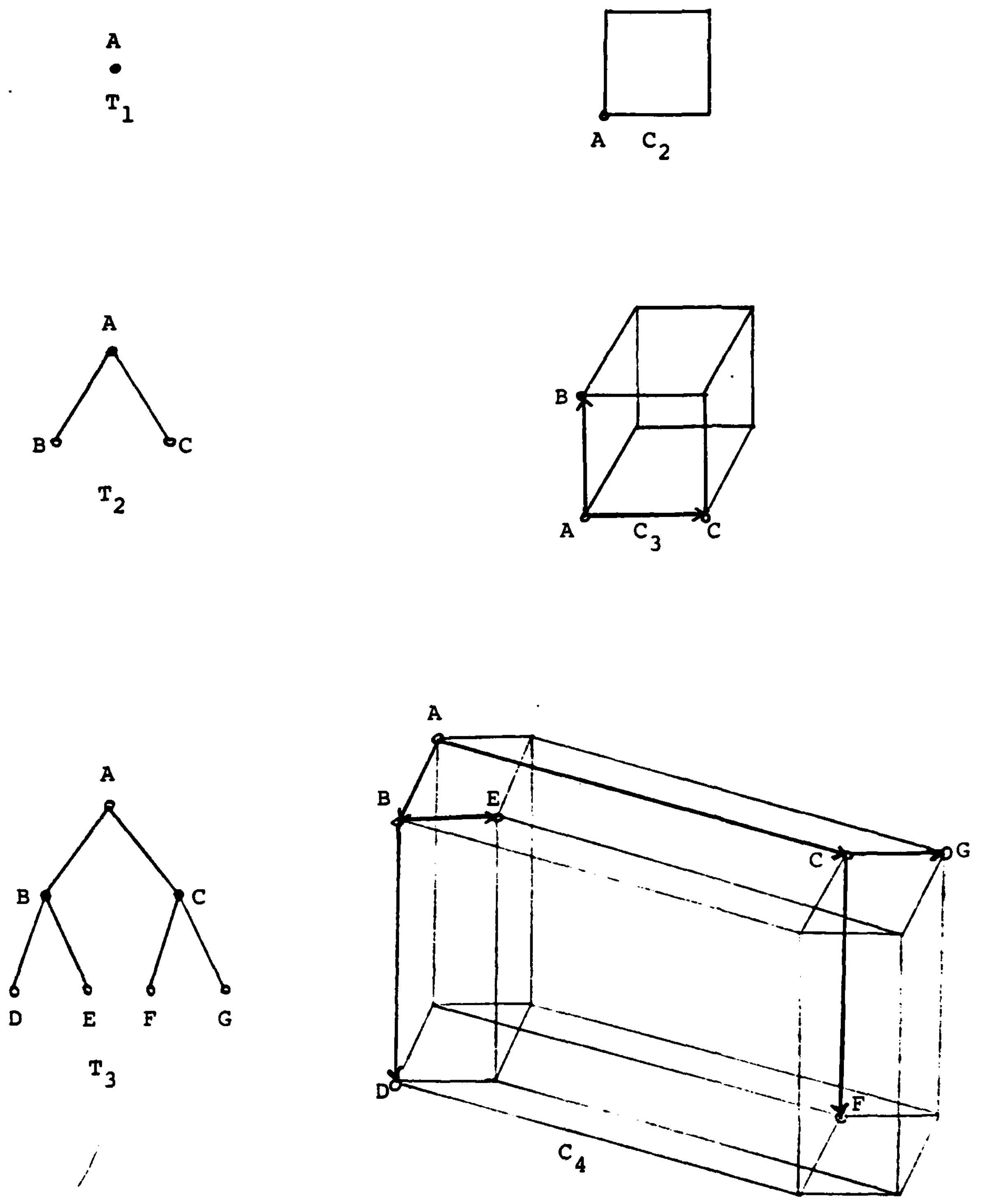

Figure 1: Embeddings of T_1, T_2, T_3 into C_2, C_3, C_4.

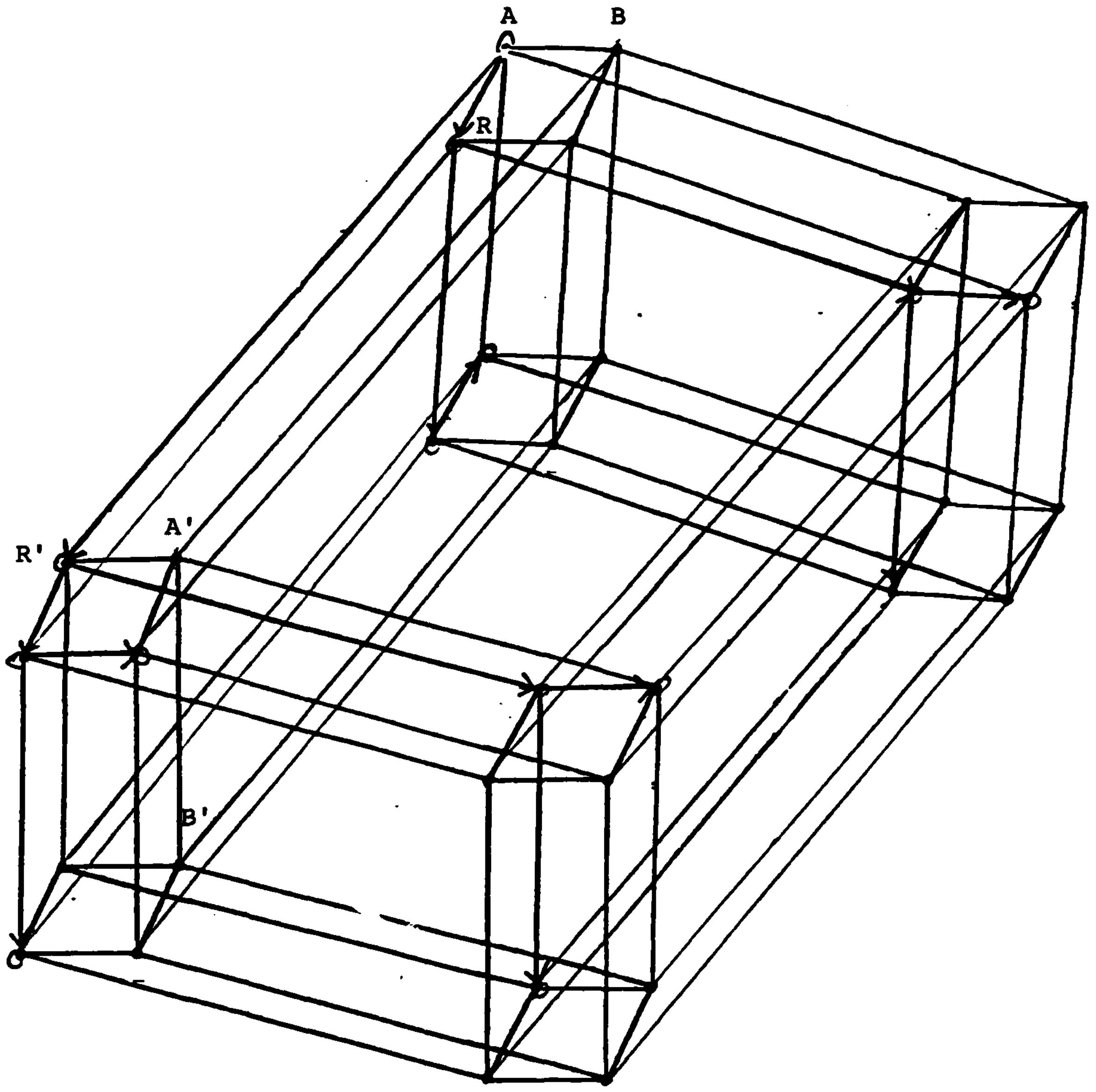

Figure 2: The free-free-neighbor property.

In C_4, A is a free neighbor of R (image of root of T_3)
B is a free neighbor of A
In C'_4, A' is a free neighbor of R' (image of root of T'_3)
B' is a free neighbor of A'.

T_3 and T'_3 are the subtrees of T_4.

In C_5, A becomes the new root of T_4,
B is a free neighbor of A,
A' is a free neighbor of B.

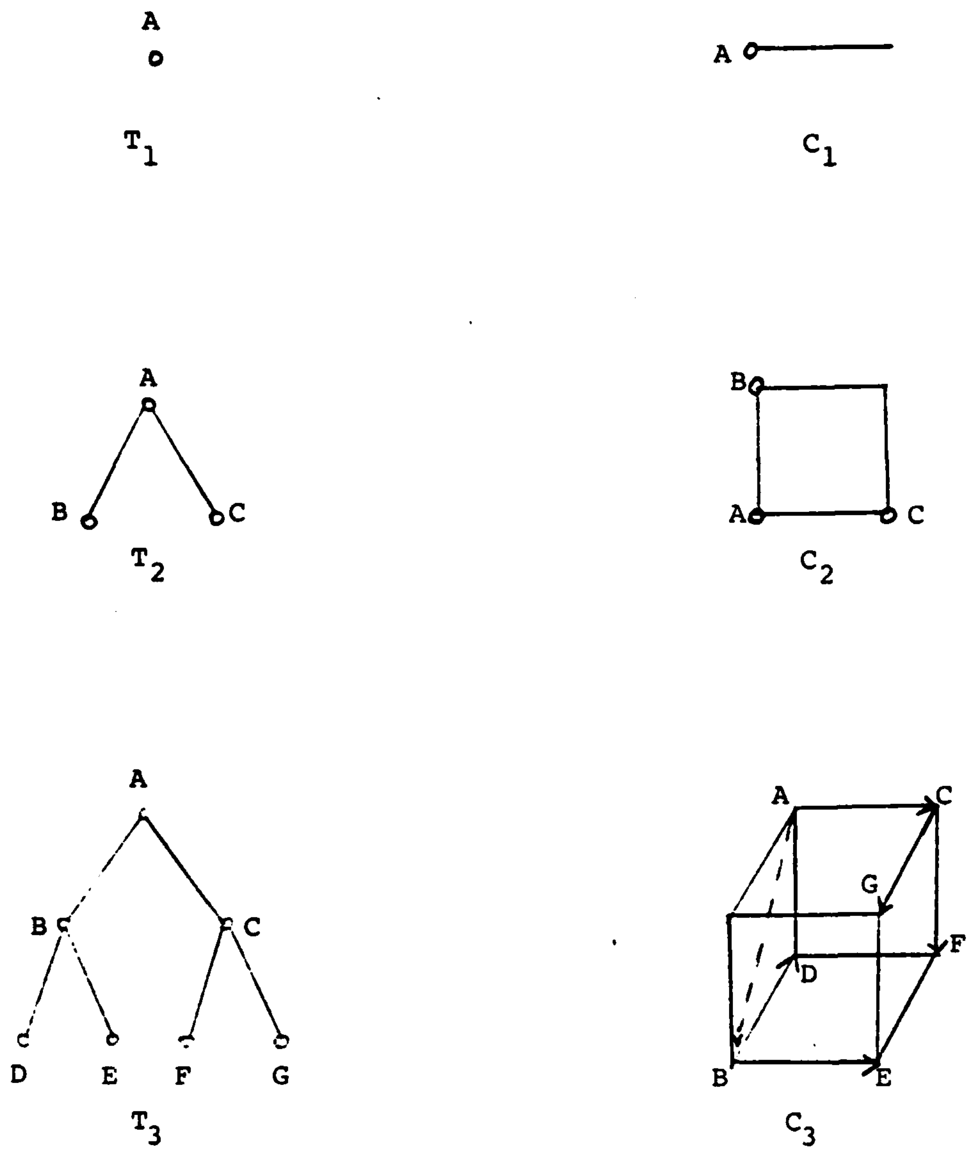

Figure 3: Embeddings of T_1, T_2, T_3 into C_1, C_2, C_3 with dilation costs of 2.
In C_3, the dotted line indicates that the distance between A and B is 2.

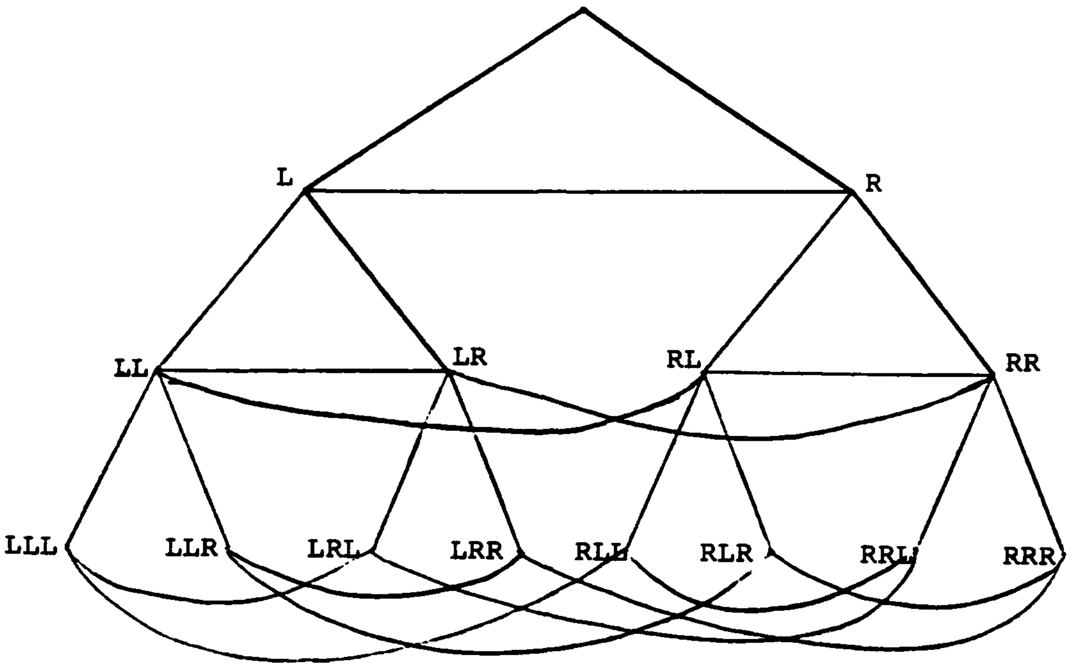

Figure 4

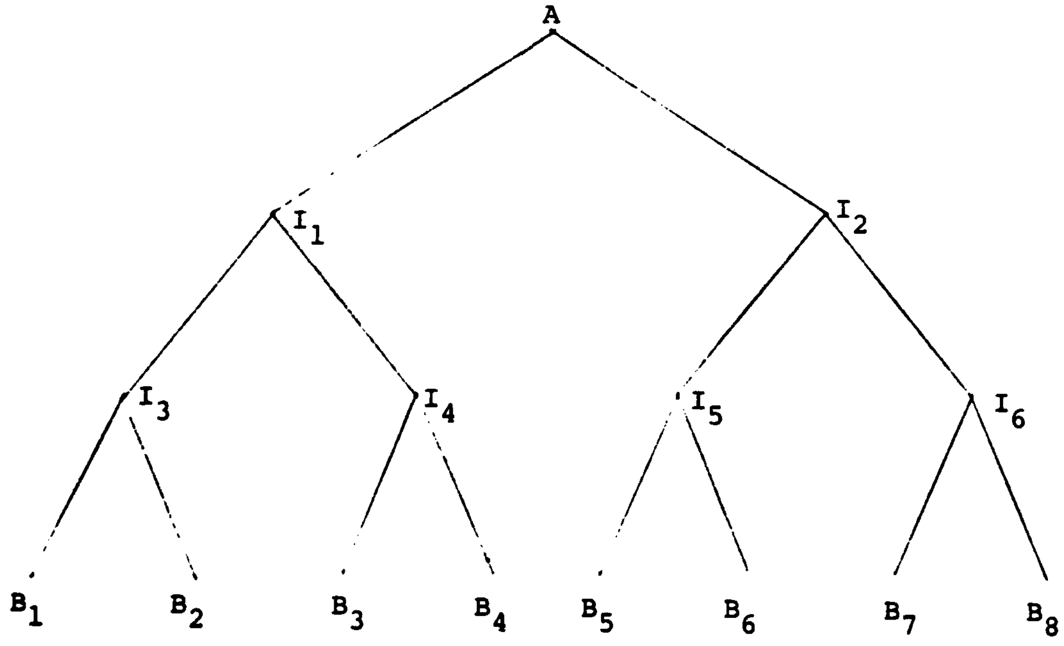

Figure 5: $B_1, B_2, \ldots, B_8$ are the children of node A in an 8-ary tree. $I_1, I_2, \ldots, I_6$ are the nodes inserted to convert the 8-ary tree to a binary tree.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 84-0175	2. GOVT ACCESSION NO. AD-A139847	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) EMBEDDING OF NETWORKS OF PROCESSORS INTO HYPERCUBES		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER CAR-TR-36 ; CS-TR-1354
7. AUTHOR(s) Angela Y. Wu		8. CONTRACT OR GRANT NUMBER(s) F49620-83-C-0082
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Automation Research University of Maryland College Park, MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Math. & Info. Sciences, AFOSR/NM Bolling AFB Washington, DC 20332		12. REPORT DATE December 1983
		13. NUMBER OF PAGES 23
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

parallel processing
networks of processors
hypercubes

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The hypercube is a good host graph for the embedding of networks of processors because of its low degree and low diameter. Graphs such as trees and arrays can be embedded into a hypercube with small dilation and expansion costs, but there are classes of graphs which can be embedded into a hypercube only with large expansion cost or large dilation cost.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)